Reinforcement Learning
Exercise 3

Q-Learning

In this the exercise, you'll be applying grid-based Q-learning to the *Cartpole* and *LunarLander* environments. You will need to modify `train.py` and `plot_value.ipynb`.

Cartpole

Recall the Cartpole environment from Exercise 1.

Task 1.1 – 25 points Implement Q-learning as presented in [1] Section 6.5 for the Cartpole environment in file `train.py`. We need to compare two exploration methods:

(a) using a constant value of $\epsilon = 0.1$;

(b) using GLIE (i.e. greedy in limit with infinite exploration) which reduces the value of ϵ over time. Its formula can be found from the lecture.

For GLIE, aim at reaching $\epsilon = 0.1$ after 20000 episodes and round the value of constant b to the nearest integer. You can run with constant ϵ by using command `python train.py epsilon=0.1` or with GLIE by using command `python train.py epsilon=glie glie_b=<insert-correct-value>`. For GLIE, the plot should be similar with Figure 1. You can record videos of the agent's performance with command `python test.py save_video=true`; these will be saved under the `results` folder. **Attach the training performance plots in terms of episode and smoothed episodic reward for both cases in your report ("ep_reward_avg" in Weights&Biases). Also include the produced `q_table.pkl` for both cases in your submission.**

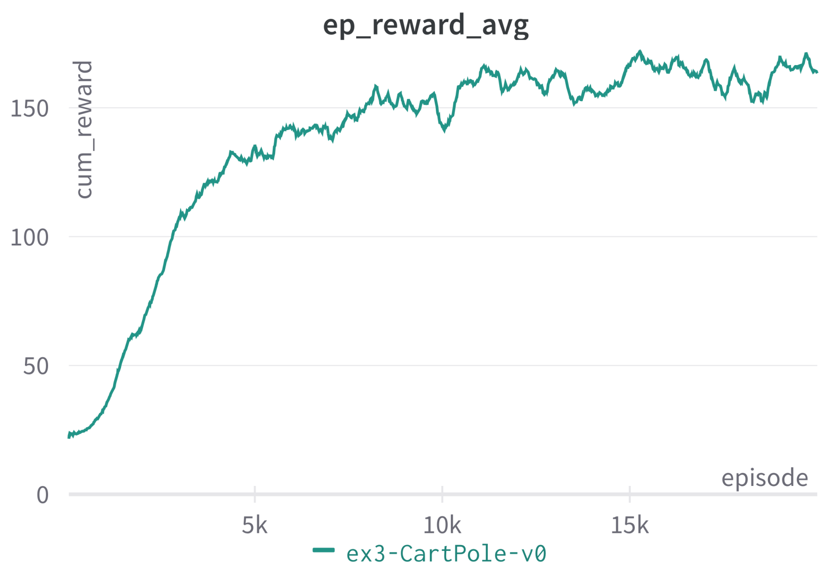

Figure 1: The training performance plot should look similar to the one presented here when using GLIE.

Task 1.2 – 10 points Use your Q-function values estimated with GLIE to calculate the optimal value function of each state. Complete the `plot_value.ipynb` to plot the heatmap of the value function in terms of x and θ, such that θ is on horizontal and x is on vertical axis. For plotting, average the values over $\dot{x}$ and $\dot{\theta}$. **Attach the heatmap in your report.**

Hint: For plotting the heatmap you can use Matplotlib-pyplot: `pyplot.imshow(values_array)` or Seaborn: `seaborn.heatmap(values_array)`

Question 1 – 15 points What do you think the heatmap would have looked like:

(a) before the training?

(b) after a single episode?

(c) halfway through the training?

Justify why for all the cases. Attaching the plots is not required.

Task 1.3 – 10 points Set ϵ to zero, effectively making the policy greedy w.r.t. current Q-value estimates. Run the training again while

(a) keeping the initial estimates of the Q function at 0,

(b) setting the initial estimates of the Q function to 50 for all states and actions.

You can change the initial Q value estimates by passing `initial_q=<value>` in the command line. **Attach training performance plots of both initializations in your report.**

Question 2 Based on the results you observed in Task 1.3, answer the following questions:

Question 2.1 – 5 points In which case does the model perform better?

Question 2.2 – 15 points Why is this the case, and how does the initialization of Q values affect exploration?

Lunar lander

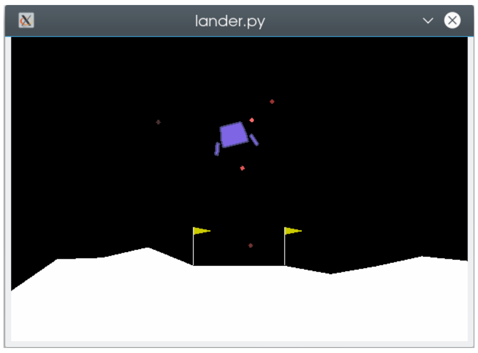

Figure 2: The Lunar lander environment.

The *Lunar lander* environment is shown in Figure 2. The goal is to make the lunar lander land on the ground between two flag poles. The agent receives a positive reward for moving towards the landing pad, landing, etc. A negative reward is given for firing the main engine (more fuel-efficient policies are better) and for crashing. Four actions are available: firing the left/right/main engines, or doing nothing (free fall). The observation vector consists of 6 continuous and 2 discrete values:

$$o = \begin{pmatrix} x & y & \dot{x} & \dot{y} & \theta & \dot{\theta} & c_l & c_r \end{pmatrix}^T, \tag{1}$$

where x and y are the coordinates of the lander, $\dot{x}$ and $\dot{y}$ its velocities, θ represents the rotation angle and $\dot{\theta}$ the angular velocity of the lander. Two discrete values c_l and c_r indicate whether the lander's legs are in contact with the ground (0 or 1).

Task 2 – 5 points Run the training for Lunar Lander environment by using `python train.py env=lunarlander_v2 epsilon=glie glie_b=<value-in-task-1.1>`. Run it for 20000 episodes (which was enough for the Cartpole to learn). **Attach the training performance plot in your report.**

Hint: The produced q_table.pkl will be a large file (around 2 Gb), so run this task on an Aalto computer if the file size causes problems. If you encounter an error `box2D is not installed`, you could try `pip install Box2D`.

Question 3 – 15 points Does the lander learn to land between the flag poles? Why/why not?

Submission

The deadline to submit the solutions through MyCourses is on Oct 3, 23:55. Example solutions will be presented during exercise sessions on Oct 10.

The report *must* include:

1. **Answers to all questions** posed in the text.
2. The **training performance plots** for each of the tasks (Task 1.1 - fixed and GLIE, Task 1.3 - for both initializations, Task 2 - Lunar Lander).
3. The **heatmap** from the end of the training (Task 1.2).

In addition to the report, you must submit as separate files:

1. `q_table.pkl`, for both constant epsilon and GLIE in Task 1.1 (you can rename the files),
2. Python code used to solve the exercises.

Do not attach the Q-values for Lunar Lander.

Please remember that not submitting a PDF report following the **Latex template** provided by us will lead to subtraction of points.

For more formatting guidelines and general tips please refer to the README.md.

If you need help or clarification solving the exercises, you are welcome to join the exercise sessions. Good luck!

References

[1] Sutton, Richard S., and Andrew G. Barto. "Reinforcement Learning: An Introduction (in progress)." London, England (2017). `http://incompleteideas.net/book/RLbook2018.pdf`